29530 DUP

UNLIMITED
UNCLASSIFIED

ILLIMITÉE
NON CLASSIFIÉE

Technical Report

Rapport technique

1988 / 08

TR-SYS-020
NRC NO. 29530

ANEVENT
AN INTERACTIVE COMPUTER MODELING PACKAGE BASED ON DISCRETE EVENT SIMULATION (UPDATE OF LTR-AN-48)

G.H. Beland, A.W. Chan
P.W.U. Graefe, and L.K. Nenonen

Division of
Mechanical Engineering

Division de
génie mécanique

National Research Council Canada
Conseil national de recherches Canada

Canada

89-03255/

DIVISION OF MECHANICAL ENGINEERING PUBLICATIONS

DM **(Division of Mechanical Engineering Report)**
Scientific and technical information considered important, complete and a lasting contribution to existing knowledge.

TR **(Technical Report)**
Information less broad in scope but a substantial contribution to existing knowledge.

CTR **(Controlled/Classified Technical Report)**
A Technical Report with controlled distribution for national security, proprietary or other reasons.

LM **(Laboratory Memorandum)**
Preliminary or exploratory information with controlled distribution.

CAT **(Calibration Analysis and Test Report)**
Information on minor laboratory projects or services.

PUBLICATIONS DE LA DIVISION DE GÉNIE MÉCANIQUE

DM **(Rapport de la Division de génie mécanique)**
Informations scientifiques et techniques jugées importantes, complètes et susceptibles de contribuer de façon durable à l'avancement des connaissances courantes.

TR **(Rapport technique)**
Informations de moindre importance, mais pouvant contribuer substantiellement à l'avancement des connaissances actuelles.

CTR **(Rapport technique à diffusion contrôlé/classifié)**
Rapport technique à diffusion contrôlée pour des raisons de sécurité nationale, de propriété intellectuelle et autres.

LM **(Mémoire de laboratoire)**
Informations préliminaires ou de nature exploratoire à diffusion contrôlée.

CAT **(Rapport d'étalonnage d'analyse et d'essai)**
Informations sur de petits projets ou des services de laboratoire.

UNLIMITED
UNCLASSIFIED

ILLIMITÉE
NON CLASSIFIÉE

ANEVENT
AN INTERACTIVE COMPUTER MODELING PACKAGE BASED ON DISCRETE EVENT SIMULATION
(UPDATE OF LTR-AN-48)

ANEVENT
UN PROGICIEL INTERACTIF DE MODÉLISATION INFORMATIQUE FONDÉ SUR LA SIMULATION D'ÉVÉNEMENTS DISCRETS
(MISE À DATE LTR-AN-48)

G.H. Beland, A.V. Chan,
P.V.U. Graefe, and L.K. Nenonen

Technical Report

Rapport technique

1988/08

TR-SYS-020
NRC No. 29530

V. Thomson, Head/Chef
Systems Laboratory/
Laboratoire des systèmes

J. Ploeg
Director/
Directeur

ABSTRACT

ANEVENT is a collection of FORTRAN based subroutines for building interactive discrete event simulation models. The overall structure of such models, both interactive and non-interactive, is described. The purposes and methods of use of the routines in the package are also given.

RÉSUMÉ

ANEVENT est un jeu de sous-routines écrites en FORTRAN destiné à la construction de modèles de simulation d'événements discrets. On décrit la structure globale de ces modèles, de type interactif ou non interactif. Le texte traite aussi du rôle des routines du progiciel et des méthodes mises en oeuvre pour les réaliser.

CONTENTS

		Page
	ABSTRACT/RÉSUMÉ	(ii)
	LIST OF FIGURES	(iv)
	LIST OF TABLES	(iv)
1.0	INTRODUCTION	1
2.0	GLOSSARY OF TERMS	2
3.0	MODEL OPERATION	4
4.0	JOURNALING	8
	4.1 AN_READLINE	8
	4.2 Enabling The Journalling	8
	4.3 Common Block Saves	9
	4.4 Recalling From A Journal File	9
	4.5 EVT002	9
5.0	COMMAND INTERPRETATION	10
6.0	GASP FILE MANIPULATION	12
7.0	PROGRAMMING CONVENTIONS AND ASSIGNMENTS	13
	7.1 Subroutine Check List	13
	7.2 Input - Output	20
	7.3 File Assignment	24
	7.4 Common Blocks	24
8.0	CREATION OF THE ANEVENT LIBRARY	29
9.0	DATA FOR COMMON BLOCK ANECOM	32
10.0	TRANSFER OF ANEVENT PACKAGE TO ANOTHER COMPUTER	35
	10.1 Random Number Generation	35
	10.2 Operator Interrupt and Time Keeping	35
	10.3 Format Descriptor $	36
	10.4 Reading of Time of Day and Date	37
11.0	BIBLIOGRAPHY	38
	DOCUMENTATION PAGE	

CONTENTS (Cont'd)

LIST OF FIGURES

Figure		Page
1	Operation of interactive discrete event models.....	5
2	Operation of non interactive discrete event models.....	7
3	Command interpretation.....	11
4	Contents of file ANECOM.COB common block declarations..	25
5	Command procedure to create library ANELIB.....	30
6	SUBLST.COM list of subroutines in library ANELIB.....	31
7	ANE.DAT data file for common block ANECOM.....	33

LIST OF TABLES

Table		
1	Routines in the ANEVENT package.....	18
2	Switch assignments.....	22
3	Input-output device category assignments.....	23
4	Common block initialization.....	34

ANEVENT
AN INTERACTIVE COMPUTER MODELING PACKAGE BASED ON DISCRETE EVENT SIMULATION

1.0 INTRODUCTION

For a number of years the Division of Mechanical Engineering of the National Research Council has been collaborating with the mining, metal production and transportation industries on modeling and simulation of industrial plants. The models were used to investigate proposed changes to plant operations or equipment, the impact of proposed plant expansions or the design of new plants or transportation systems.

These plants or systems were modeled as either discrete processes or as combinations of discrete and continuous processes. For the latter type of models a hybrid computer was used. Most of the models included interactive features, permitting model operators to observe the evolution of the model state with time and to issue commands to them, thus directing the simulated behavior of the plants being modeled.

The ANEVENT simulation package has evolved from these activities and is based on the simulation language GASP II (Pritsker, et al. 1969). ANEVENT provides a structure for building interactive, discrete event computer models and provides a number of problem independent subroutines to support building of models.

This document describes the overall structure of such discrete event models, interactive or otherwise, and the purpose and use of all routines available with the ANEVENT simulation package.

For information on how to build interactive discrete event models, the reader is referred to 'ANEVENT: An Interactive Computer Modeling Package Based on Discrete Event Simulation - A User's Guide' (Crate, 1982) which describes the modeling steps necessary and illustrates the entire model building procedure by means of the development of a sample model.

A license for the program can be obtained for a moderate charge by writing to:

Dr. V.J. Thomson
Head, Systems Laboratory
Div. of Mechanical Engineering
National Research Council Canada
Montreal Road
Ottawa, Ontario
K1A 0R6

2.0 GLOSSARY OF TERMS

To facilitate discussions in the remainder of this document, the following terms which are used quite frequently are defined.

Model The real process or system is analyzed and an appropriate description is realized in the form of a SYSTEM MODEL. This SYSTEM MODEL is refined and when it is deemed to adequately describe reality for the purpose for which the model is needed, it can then be implemented on a computer. The programming specifications required for this implementation is referred to as the COMPUTER MODEL. In this document, unless otherwise specified, the term MODEL refers to the computer model.

Discrete Event Model A discrete event model represents behavior in terms of dependent variables that change discretely, at specified instances of simulated time.

Event An event is a discrete change in the status of the system being modeled.

Event Time The time at which an event occurs is called the event time.

Synchronous Mode In the synchronous mode of model operation, real time intervals between event times are simulated via time intervals that are linearly related to the real time intervals through a specified speed-up factor.

Asynchronous Mode In the asynchronous mode of model operation, simulated time is instantaneously advanced from one event time to the next.

Entity A real process or system can be described in terms of a number of entities, whose interaction influences the process or system behavior. Entities may be 'devices' such as elevators, trains, furnaces etc. or 'materials' such as ore, liquid steel, crates or passengers.

Attribute Each entity is described by attributes. For example, device attributes can describe the state of the device, (idle, busy, under repair, etc.), the time at which it assumed the current status, the size, location, weight or temperature of the device. Similarly material attributes may describe the grade, specific gravity, temperature and dimension of a unit of the material (e.g. weight of a full ladle).

Event attributes identify the type of event, event time and other parameters required in executing the event. For example, an event "An elevator arrives at a floor" could have the following attributes: (1) event time, (2) event identification, (3) elevator number, (4) floor number, (5) up/down indicator. One of the elevator attributes would be its status, possible states being "Stopped, going up", "Stopped, going down", "Ascending", etc.

Files Apart from data files prepared externally and read by the model program in order to initialize the model variables, two other types of files are used in the ANEVENT package, GASP files and Model State files. GASP files are used to store variables, appropriate attributes of the file entries being used for automatic ordering of these entries in the file according to predefined rules.

Model State files are disk files created by the program to save the entire model state. This capability allows later initialization of the model to any one of these saved states.

Time File GASP file number one is the time file and stores one entry for every event that has been scheduled. The entries are stored in chronological order of event times.

File Entries Each GASP file can store a pre-specified number of file entries, each entry being described by a number of attributes.

Ranking of File Entries Ranking concerns the means of ordering entries in a GASP file. For each file, a ranking attribute and a ranking rule is specified. For example, the entries in the time file (file 1) are ranked on attribute 1, the event-time, using the low-value-first (LVF) ranking rule, thus ensuring that the first entry is the next event to be executed. For each of the other GASP files, the user can choose any attribute for ranking and one of two ranking rules, low-value-first or high-value-first (HVF).

GASP General Activity Simulation Program, a FORTRAN based discrete event simulation language developed at US STEEL. Extensions to this language resulted in GASP II and GASP IIA (Pritsker, et al. 1969) and the combined discrete-continuous simulation language GASP IV (Pritsker 1974). ANEVENT uses the GASP II file manipulation mechanism.

3.0 MODEL OPERATION

The user can choose to run his model in the synchronous or asynchronous modes by specifying the value of a model mode variable.

In the synchronous mode, the system state is often displayed through CRT messages or graphics images that are updated at intervals specified by the model user. This mode is used to evaluate operating strategies in detail in an interactive fashion, with most of the decision-making being done by the model operator.

The asynchronous mode is usually selected in cases where all decision making is built into the model and the user is interested in production or performance statistics for long periods of simulated time. Display of the model state during simulation is usually not requested. However, periodic snapshots of the model state can be stored on a disk for later examination. The model mode may be changed during a simulation ran under operator or program control.

The operation of interactive discrete event models using the ANEVENT package is illustrated in figure 1 and consists of the following steps.

1. Model initialization sets all model variables and parameters to reflect the initial state of the system. It includes the filing of entries into the time file (GASP file 1) for those time events whose event times are known at the start of a simulation run. At least one time event has to be filed, and since the events filed are stored in chronological order the first entry corresponds to the event to be executed first.

2. The time at which the next event is to be executed is set to the event time of the event at the top of the time file.

3. If the simulation mode is synchronous, proceed with step 4, otherwise proceed directly with step 7.

4. A computer clock is set to generate a timer interrupt after the interval of time to the event time of the next event has elapsed. This interval is the time difference between the next event time and the current simulation time, suitably scaled by a specified speed-up factor.

5. The program suspends further action until either a timer interrupt or a model operator interrupt occurs.

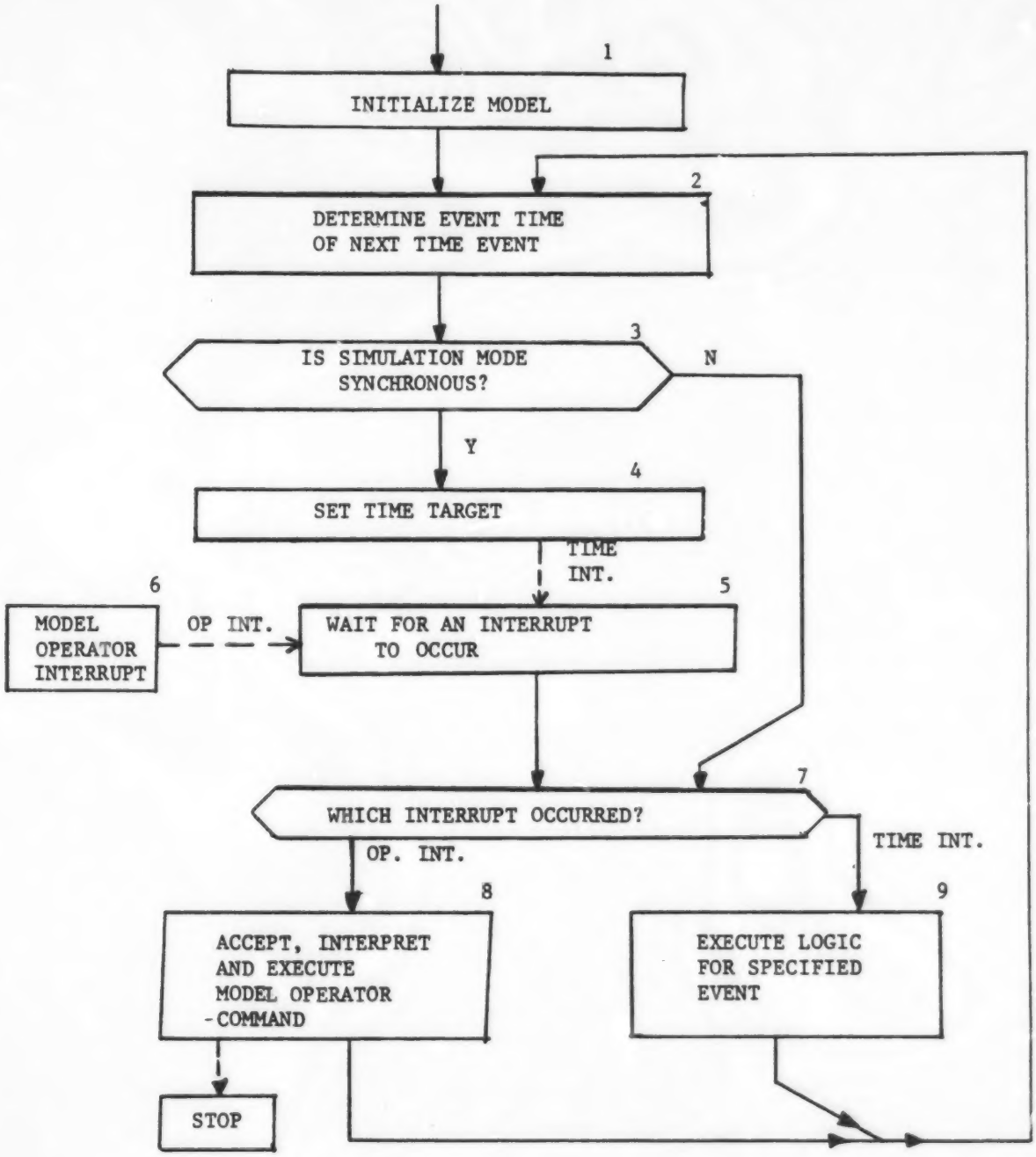

Figure 1. Operation of interactive discrete event models

6. The model operator can interrupt the simulation process at any time by pressing a special key (the RETURN key in the VAX version) in order to issue commands to the model.

7. The type of interrupt that has occurred is examined. If an operator interrupt was detected, proceed with 8. For a timer interrupt or both interrupts occurring simultaneously proceed with 9.

8. The model operator is prompted for input, his commands are accepted, interpreted and executed. Depending on the nature of each command, execution may entail an update of model variables or parameters, a printout of information, the filing of new events into the time file or terminating the simulation run. On completion of the command input session, proceed with 2 unless the simulation run has been terminated by a QUIT command from the operator.

9. The logic of the specified event is executed. Inputs to the event routine are provided by the event attributes and possibly other variables or parameters of the data base. On completion of the event execution, proceed with 2. Note that for the case of simultaneous interrupts from the timer and operator only the timer interrupt has been serviced. The operator interrupt will be serviced only after control has again passed through steps 2 to 7.

Note that the model clock does not run while an event is being executed or operator commands are being processed. During these times, model time remains frozen.

Some models require no operator intervention and are run in the asynchronous mode. For these models, the model structure is simplified considerably in that steps 3 to 8 are not required [Fig. 2]. In this case, a simulation run is terminated via an end-of-simulation event which is filed in the time file at the start of simulation, it's event time being based on run length specifications provided during initialization.

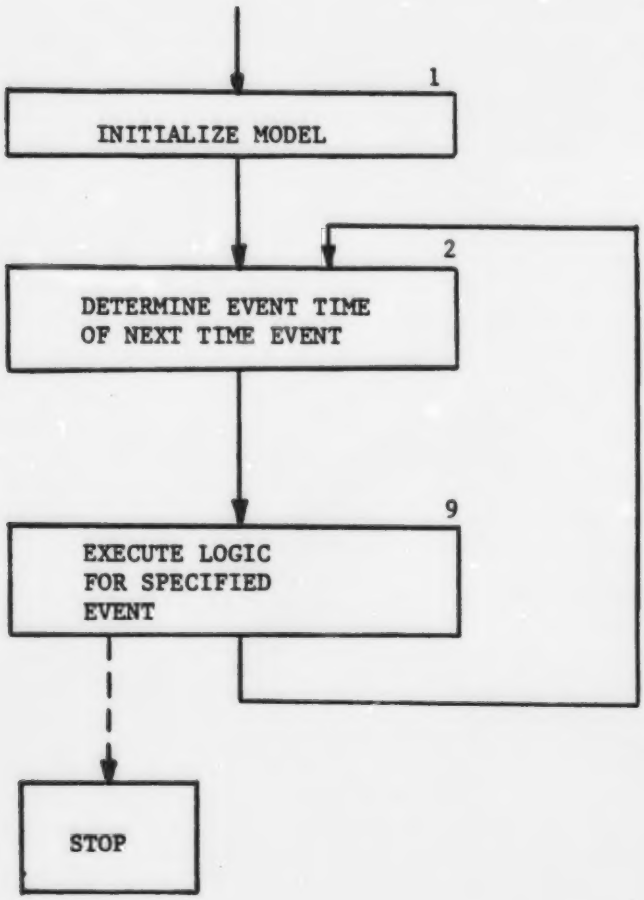

Figure 2. Operation of non interactive discrete event models

4.0 JOURNALING

The ANEVENT package has a journaling capability allowing the user to recall a model from any simulation time of a previous run. Common block saves are done periodically and all user interaction is recorded in a journal file. When a recall is requested, the common-block save file which is prior and most recent to the recall time is read to initialize the common blocks. Any commands in the journal file entered by the user after the common block save was done are executed until the recall time is reached.

The journaling capability can also be used for training purposes. If an ANEVENT switch is set (AN_SWITCH(5)) the model will pause when recalling a command from a journal file until the operator types a carriage return. This controlled 'play-back' mode allows the operator to 'learn' the commands as they are executed.

4.1 AN_READLINE

A function called AN_READLINE handles all reading for the model. Any time the model accepts something from the keyboard, this function is called instead of the FORTRAN READ statement. This must be used for journaling to work. The calling sequence is:

```
IF (.NOT.AN_READLINE(VAR,LINE)) RETURN
```

The function returns TRUE or FALSE depending on whether the read was successful. VAR is a real array of 10 elements and LINE is a character string of 80 characters. The function reads either from the keyboard or from an input journal file, depending on whether the model state is being recalled from an input journal file or not. Any numbers found on the input line are put into VAR as they are encountered. The complete input line is returned in LINE and its length in LINE_LEN.

4.2 Enabling The Journaling

Routine AN_INITIA sets up the journalling for ANEVENT. It prompts the user for a warm or cold start. If the user specifies a cold start, AN_COMINT and AN_SUBINT are called to initialize the common blocks. For a warm start, the user is asked for the input journal filename and the recall time. AN_RECALL initializes the common blocks to the specified time. The user is also asked if an output journal file is to be created for this simulation run. If so, the user is prompted for the filename and the time interval between system saves. Common block saves are performed by EVT001 which reschedules itself for the time of the next save using the interval the user has entered.

4.3 Common Block Saves

EVT001 calls AN_SAVEST to save all common blocks in the model. It is initially filed by AN_INITIA and refiles itself at the interval specified in AN_INITIA.

4.4 Recalling From A Journal File

AN_RECALL recalls the model state from a journal file for the time specified in AN_INITIA. When journalling is enabled, all user interaction with the keyboard is recorded in a journal file along with markers to indicate the system saves and operator interrupts. AN_RECALL will use the system save most recent and prior to the recall time to initialize the common blocks and then will execute the remaining commands in the journal file until it reaches the recall time.

4.5 EVT002

This event generates an operator interrupt and is used when recalling from a journal file. When a user is running a model with the journaling turned on, all operator interrupts are saved in the journal file. If a recall is done on this run, AN_RECALL will file EVT002 for the appropriate times to generate the same interrupts and allow the model to accept commands from the journal file. AN_RECALL also files this event to signal the end of the recalling. This event can be used to schedule one or more interrupts for stopping the model at given points in time during a simulation.

5.0 COMMAND INTERPRETATION

Operator command interpretation is done at two levels as illustrated in Figure 3. At the first level, the operator is asked to select one of these three command categories:

USER
DISPLAY
MISCELLANEOUS

Once a category has been chosen, the operator is prompted for a command in that category until a carriage return signals the end of commands for that category. Then, control is returned to the first level which prompts for the next command category. If no further commands are to be issued, another carriage return terminates the command input session.

The USER category is the responsibility of the user who must add appropriate code in routine AN_COMTYP to process desired user commands.

The DISPLAY category has three possible commands:

1 Turn on display
2 Turn off display
3 Change update interval for display

The MISCELLANEOUS category has the following commands: CHANGE, REMOVE, INFO, ADD, SWITCH, SCHEDULE and QUIT. Each miscellaneous command invokes an associated routine AN_COMMAND i.e. AN_CHANGE. See the description of associated routine for the function of each command.

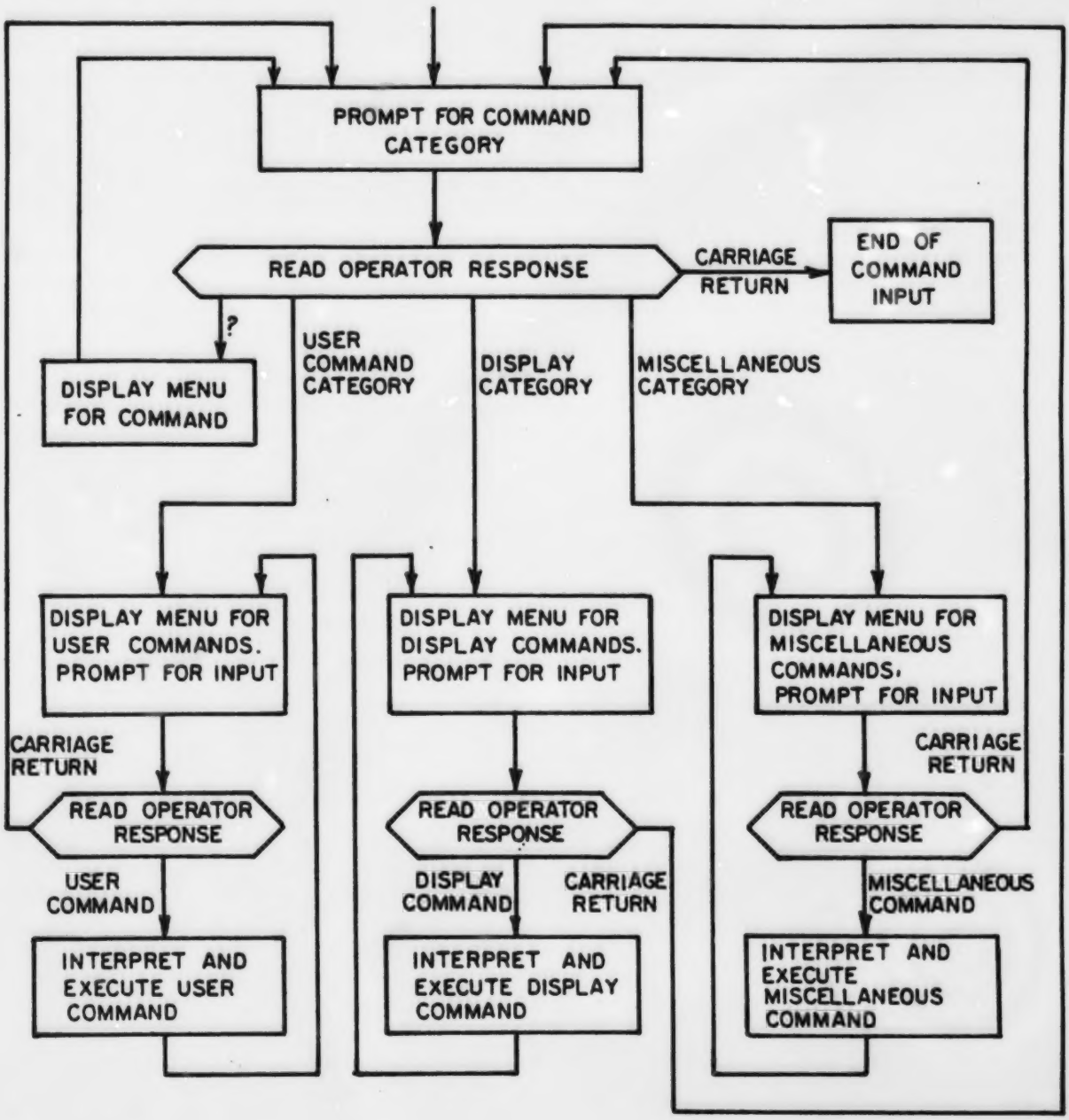

Figure 3. Command interpretation

6.0 GASP FILE MANIPULATION

The GASP files use a linked list structure described in detail by Pritsker and Kiviat (1969) on the GASP II simulation language. A two dimensional array NSET of IMZZ+2 rows and IDZZ columns represents the filing space, shared by all files. Each file entry occupies one column which provides for the storage of IMZZ entry attributes as well as a pointer to the column where the successor entry is stored and a pointer to the column where the predecessor is stored. For each file there is a pointer to the column number for the first entry, MFE and the last entry, MLE. Array KRANK stores the ranking attributes for each file and array INN specifies the ranking rule. An array NQZZ keeps track of the number of entries in each file. Entries are entered into the file by loading entry attributes into a transfer vector ATRIB and then calling the filing routine AN_FILEM(NFILE) to file the entry into file NFILE. A call to AN_RMOVE removes an entry and copies entry attributes into array ATRIB. The following file manipulation routines are available: AN_DATAN, AN_FILEM, AN_INSERT, AN_RMOVE, AN_DEFILE, AN_DSTROY, AN_FIND, AN_FIND2, AN_FIND3.

7.0 PROGRAMMING CONVENTIONS AND ASSIGNMENTS

In this section, program organization, conventions and assignments are discussed.

7.1 Subroutine Check List

Table 1 provides an alphabetical list of all routines in the ANEVENT package. It indicates which routines are required for the interactive version and the non-interactive version, which routines are utilities, which ones are complete and which require some user generated code. The following list summarizes the purpose of each routine, the arguments required and, where applicable, the required function of the user code to be added to the routine.

ANEVENT — Main program for the interactive model, controls simulation run.

ANEVENT1 — Main program for the non-interactive model, controls simulation run.

AN_ADD — Responds to the add command 'ADD,NFILE' to add an entry to file NFILE.

AN_CHANGE — Responds to the 'CHANGE,NFILE,NENTRY' command to make changes to entry NENTRY of GASP file NFILE.

AN_CMDERR — Provides printouts of errors during command decoding.

AN_CMNDRD — Accepts commands from the model operator.

AN_COMINT — Initializes common block ANECOM except for I/O parameters which were initialized in ANEVENT or ANEVENT1.

AN_COMTYP — Processes model operator commands according to type by calling AN_MISCOM for Miscellaneous commands, changing display for the DISPLAY command or executing user-provided code for User-defined commands. The user must add coding following label 100 to process the user commands.

AN_CURPOS — Stores or restores cursor position on screen.

AN_DATAN — Initializes all GASP filing parameters and some GASP file entries.

AN_DEFILE(NFILE,NVAL,JATT)
Removes all entries from file NFILE with the value NVAL in the JATT'th attribute.

AN_DIST(NUM)
Returns a value given the distribution number NUM. Five types of distributions are currently provided: constant, uniform, exponential, normal and Weibull.

AN_DSTROY(NFILE)
Lists and then deletes all entries of GASP file NFILE.

AN_EVNTS Passes control to the routine whose event number is AN_EVENT.

AN_EVT001 Handles the common block saves and files itself for the next save using the interval specified in AN_INITIA.

AN_EVT002 Generates an operator interrupt and is used when recalling from a journal file.

AN_EVT003 Updates alphanumeric display.

AN_FILEM(NFILE)
Files the content of array ATRIB as an entry of GASP file NFILE.

AN_FIND(NVAL,NFILE,JATT,KCOL,IBF)
Attempts to find the first entry in GASP file NFILE whose JATT'th attribute has the value NVAL. If such an entry is found, KCOL is set to the column number of the filing array where the entry is stored, and a copy of the entry is made in array IBF. If no such entry is found, KCOL is returned as zero.

AN_FIND2(NVAL1,NVAL2,NFILE,JATT1,JATT2,KCOL,IBF)
Is similar to routine AN_FIND except that the search includes two values NVAL1 and NVAL2 as the JATT1'th and JATT2'th attribute.

AN_FIND3(NVAL1,NVAL2,NVAL3,NFILE,JATT1,JATT2,JATT3,KCOL,IBF)
Is similar to routine AN_FIND except that here, three values and three attributes are involved.

AN_HEADER(ID)
Advances the output device whose code is given by ID to a new page, and prints a user-specified header as well as the time and date at which the simulation run was started.

AN_INFO Decodes the INFO command: INFO,CATEGORY where CATEGORY is GASP or HISTOGRAM.

AN_INITIA Initializes the model from either data files or model state files.

AN_INSCRN(IU)
Initializes the display screen unit IU.

AN_INSERT(NFILE,KCOL,IFL)
Inserts the entry stored in array AN_ATRIB into file NFILE. For IFL=1 the entry is stored logically before the entry in column KCOL. For IFL=2 the entry is stored logically after the entry in column KCOL.

AN_IRNG(IX,KUMX,NCX,ISTRM,IOUT)
Returns a random deviate IOUT of stream ISTRM generated from the NCX points of the cumulative distribution with co-ordinates stored in arrays IX and KUMX. IOUT takes on values of elements of IX. See Phan (1979) for preparation of data for IX,KUMX,NCX.

AN_LISTFL(NFILE)
lists in logical order the entries of GASP file NFILE.

AN_MISCOM Decodes and executes miscellaneous commands. These commands are CHANGE, REMOVE, INFO, ADD, SWITCH, SCHEDULE and QUIT.

AN_MODRUN Starts the model clock and waits for an interrupt. When an interrupt occurs, AN_FLAG is set to 1 for an operator interrupt or to 2 for a timer interrupt. The interrupt is cleared and simulation time AN_SIMTIM is computed.
In the asynchronous mode of operation (AN_MODLMOD=2) setting of the clock and waiting for an interrupt is bypassed. Only the operator interrupt is examined, the timer interrupt is assumed to have occurred in all cases.

AN_QUIT Terminates the simulation run by releasing the flag cluster for the timer and operator interrupts and then stopping. The user may insert further coding in this routine to process problem-specific tasks at the termination of the run.

AN_RAN1(ISTRM)
This function routine generates a uniformly distributed random number AN_RAN1 between 0. and 1. from random number stream ISTRM.

AN_RECALL Recalls the model state from a journal file. It uses the system save most recent to the recall time to initialize the common blocks and then will execute the remaining commands in the journal file until it reaches the recall time.

AN_READLINE(VAR,LINE)
Reads the next line either from the keyboard or from an input journal file. Any numbers found on the input line are placed into VAR. The complete input line is returned in LINE and it's length in LINE_LEN.

AN_REMOVE Responds to the command 'REMOVE,NFILE,NENTRY' by removing the NENTRY'th entry of file NFILE.

AN_RMOVE(KCOL,NFILE)
Removes from GASP file NFILE the entry stored in column KCOL of the filing array and copies it into the array AN_ATRIB.

AN_RN15(IRN)
Using IRN as a positive odd integer between 1 and 32767 as a random number seed this routine will generate another uniformly distributed positive odd random number IRN.

AN_SAVEST Saves the model state in the model state file whose number is given by AN_FLAG. The user has to provide a value for variable AN_LUSCOM, the combined length of all user defined common blocks in short words (2 bytes).

AN_SET(NFILE)
Updates all relevant pointers in the GASP file manipulation system in response to a filing, insertion or removal of an entry for file NFILE. The filing array is initialized by this routine if common variable AN_INIT is set to 1.

AN_SETIME Sets the computer clock to an interval corresponding to AN_TNEXT - AN_TLAST appropriately scaled by the speedup factor AN_SPEEDUP and the conversion factor for model time units and minutes, AN_TUPM. The maximum interval that can be set is specified by AN_TIMAX.

AN_SETSW Allows the model operator to examine the current status of all 10 ANEVENT switches and to set or reset any or all of these switches.

AN_SIMDAY(ITIME,IDAY,IHOUR,IMIN,ISEC)
Computes simulation time of IDAY days, IHOUR hours, IMIN minutes and ISEC seconds from ITIME, taking into account any hour and minute offsets AN_HROFST and AN_MOFST.

AN_SUBINT Initializes all submodels.

AN_TOSCRN(IU,ITYPE,IROW,ICOL)
Positions cursor at line IROW and column ICOL on screen for output on unit IU,
ITYPE = 0 to write
= 1 to blink
= 2 to display reverse video.

AN_W2(XMEAN,DEVIAT,ALFA1,BETA1)
Computes Weibul parameters based on mean/deviation of distribution.

Routine Name	Required for Int.	Required for Non-Int.	Utility Program	Complete	User Code Required
ANEVENT	X			X	
ANEVENT1		X		X	
AN_ADD	X			X	
AN_CHANGE	X			X	
AN_CMDERR	X			X	
AN_CMNDRD	X			X	
AN_COMINT	X	X		X	
AN_COMTYP	X				X
AN_CURPOS	X	X	X	X	
AN_DATAN	X	X		X	
AN_DEFILE	X	X	X	X	
AN_DIST	X	X	X	X	
AN_DSTROY	X	X	X	X	
AN_EVNTS	X	X			X
AN_EVT001	X	X		X	
AN_EVT002	X			X	
AN_EVT003	X	X			X
AN_FILEM	X	X	X	X	
AN_FIND	X	X	X	X	
AN_FIND2	X	X	X	X	
AN_FIND3	X	X	X	X	
AN_HEADER	X	X	X		X

Table 1. Routines in the ANEVENT package

Routine Name	Routines for Interrupt-Int.	Utility Program	Complete	User Code Required
AN_[illegible]	X			X
AN_INI[illegible]	X	X	X	
AN_INSCRN	X	X		X
AN_INSERT	X	X	X	
AN_IRNG	X		X	
AN_LISTFL	X	X	X	
AN_MISCOM	X		X	
AN_MODRUN	X		X	
AN_QUIT	X			X
AN_RAN1	X	X	X	X
AN_RECALL	X			X
AN_READLINE	X		X	X
AN_REMOVE	X			X
AN_RMOVE	X	X	X	X
AN_RN15	X	X	X	X
AN_SAVEST	X			X
AN_SET	X	X		X
AN_SETIME	X			X
AN_SETSW	X			X
AN_SIMDAY	X	X	X	X
AN_SUBINT	X	X		X
AN_TOSCRN	X	X	X	X
AN_W2	X	X	X	X

Table 1 (Cont'd) Routines in the ANEVENT package

7.2 Input - Output

At run time, the user can specify input and output devices for broad categories of inputs and outputs. 20 such categories have been provided through the use of variables PRINT1 to PRINT20 to be used as device specifications for input and output. These variables are equivalenced to array AN_DEVICE in common block ANECOM and thus can be set during initialization or during a simulation run.

Ten logical variables in the form of array AN_SWITCH in common block ANECOM are provided to control program logic via the settings of these 'switches'. A typical use of switches and I/O device specification is the following:

```
IF(AN_SWITCH(3)) WRITE(PRINT5,100) I,J
```

This provides a display of variables I and J on display device PRINT5 if switch 3 is set. If PRINT5 is a CRT terminal, for instance, and the user wants to have a hard copy on a line printer, he only has to change the value of PRINT5 appropriately.

Some of the input/output categories and switches have already been assigned as shown in Tables 2 and 3. To further illustrate the use of I/O device assignments, consider the following example. Suppose that on the user's computer system, output device numbers are as follows:

```
Operator keyboard                 device 5
Operator CRT display              device 6

Disk file with name FOR003.DAT device 3
Disk file with name FOR007.DAT device 7
```

The model operator assigns values to AN_DEVICE (and so to PRINT1, etc.) by entering appropriate values into dataset ANE.DAT. Specifically, to get error messages on the CRT display he sets

```
AN_DEVICE(1) = PRINT1 = 6
```

To get a log of filing array manipulation on the disk file FOR003.DAT he sets

```
AN_DEVICE(2) = PRINT2 = 3
```

To obtain on echo, output of his input data on the same disk file he sets

```
AN_DEVICE(3) = PRINT3 = 3
```

Information display is desired on the CRT screen, hence

```
AN_DEVICE(5) = PRINT5 = 6
```

Prompts for commands are also desired on the screen, hence

```
AN_DEVICE(6) = PRINT6 = 6
```

Operator commands are to be entered on the keyboard, hence

```
AN_DEVICE(19) = PRINT19 = 5
```

If the operator later decides that error messages should come out on the disk file FOR007.DAT he simply initializes AN_DEVICE(1) to 7 instead of 6.

SWITCH #	ACTION IF SWITCH IS SET
1	Display activity, file number and attributes when filing, inserting or removing entries of a GASP file
3	Display event number and simulation time prior to event execution
4	Echo input data
5	Pause when recalling a command from a journal file until the operator presses a key
10	Turn on Alphanumeric Display

Table 2. Switch assignments

DEVICE CATEGORY I AS SPECIFIED BY AN_DEVICE(I)	CATEGORY DESCRIPTION
1	Error messages due to invalid event numbers, filing array full, or the occasion of resetting the simulation time.
2	Filing array manipulation such as filing, inserting, removing, listing of contents.
3	Echo of input data
5	Information display
6	Prompts for operator commands
9	Input journal file
10	Output journal file
19	Acceptance of operator commands

Table 3. Input/output device category assignments

7.3 File Assignment

Only the time file has been assigned in this program package. GASP file 1 is the time file. A total of NOQYY GASP files have been provided, allowing the user to assign files 2 to NOQYY as required.

Up to IMYY attributes have been allowed for each file entry. For the time file, event time has been assigned to attribute 1 and event number to attribute 2. Assignment of attributes 3 to IMYY are at the user's disposal, as are all attributes for user defined GASP files.

7.4 Common Blocks

The ANEVENT variables and arrays are stored in a system common block ANECOM. Parameter specifications, variable type declarations, dimensioning and equivalencing are included with the common block statements in file ANECOM.COB which is included in most subroutines of the ANEVENT package. The file also contains another common block called JOURNAL which is used to store the input and output journal file names in variables IN_JRNL and OUT_JRNL. A listing of this file is shown in figure 4.

The user must provide a similar .COB file for each user-defined common block. This file should also contain a parameter statement to set parameter LUSCOM to the combined length of all user-defined common blocks in 16-bit words. Apart from user-written routines, all ANEVENT package routines which require user additions may require the inclusion of these user-defined common blocks.

Following is a description of all parameters and variables used by the ANECOM common block.

PARAMETERS

IMYY	Maximum number of file attributes
IDYY	Maximum number of entries in the GASP filing array
NOQYY	Maximum number of GASP files
LANECO	Size of ANECOM common block in 16-bit words

```
C
C
C --- ANEVENT system common block: ANECOM.COB
C
C
C       Parameters are:
C
C       IMYY    -- maximum number of attributes
C       IDYY    -- maximum number of entries in filing array
C       NOQYY   -- maximum number of gasp files
C       LANECO  -- length of ANECOM common block in 16-bit words
C
        PARAMETER IMYY = 10,IDYY = 500,NOQYY = 50,LANECO = 13270
C
C
        LOGICAL AN_SWITCH,AN_READLINE
C
        CHARACTER       IN_JRNL*13,OUT_JRNL*13,AN_INPBUF*80
C
        REAL    AN_DATE,AN_TIME,AN_TUPM,AN_DIST_PAR,AN_DIST
C
        COMMON /JOURNAL/ IN_JRNL,OUT_JRNL
C
        COMMON /ANECOM/ AN_OPFLAG,AN_TIMFLG,AN_CHAN,
        1       AN_IDZZ,AN_IMZZ,AN_INIT,AN_MFA,AN_MXZZ,AN_NOQ,
        1       AN_ISEED,AN_MXX,AN_OUT,AN_ATRIB(IMYY),AN_INN(NOQYY),
        1       AN_KRANK(NOQYY),AN_MFE(NOQYY),AN_MLC(NOQYY),
        1       AN_MLE(NOQYY),AN_NQZZ(NOQYY),AN_TATRIB(IMYY),
        2       AN_NSET(12,500),AN_HROFST,AN_MOFST,
        3       AN_TLAST,AN_TNEXT,AN_TIMAX,AN_SIMTIM,AN_SPEEDUP,
        4       AN_TUPM,AN_EVENT,AN_ERR,AN_FLAG,AN_MDLMOD,AN_DATE(3),
        4       AN_TIME(2),AN_SWITCH(10),AN_DEVICE(20),
        6       AN_INPBUF,AN_WORK(IMYY),
        8       AN_SEED(100),LINE_LEN,AN_DIST_PAR(25,5)
C
        EQUIVALENCE (AN_DEVICE(1),PRINT1),(AN_DEVICE(2),PRINT2),
        1           (AN_DEVICE(3),PRINT3),(AN_DEVICE(4),PRINT4),
        2           (AN_DEVICE(5),PRINT5),(AN_DEVICE(6),PRINT6),
        3           (AN_DEVICE(7),PRINT7),(AN_DEVICE(8),PRINT8),
        4           (AN_DEVICE(9),PRINT9),(AN_DEVICE(10),PRINT10),
        5           (AN_DEVICE(11),PRINT11),(AN_DEVICE(12),PRINT12),
        6           (AN_DEVICE(13),PRINT13),(AN_DEVICE(14),PRINT14),
        7           (AN_DEVICE(15),PRINT15),(AN_DEVICE(16),PRINT16),
        8           (AN_DEVICE(17),PRINT17),(AN_DEVICE(18),PRINT18),
        9           (AN_DEVICE(19),PRINT19),(AN_DEVICE(20),PRINT20)
```

Figure 4. Contents of file ANECOM.COB common block declarations

VARIABLES

AN_OPFLAG	VAX event flag used to indicate an operator interrupt.
AN_TIMFLG	VAX event flag used to indicate a timer interrupt.
AN_CHAN	The channel number assigned by the VAX system services routine QIO to the terminal from which the model operator issues commands.
AN_IDZZ	Maximum number of entries to be used in the filing array.
AN_IMZZ	Maximum number of attributes used for a file entry.
AN_INIT	Used for filing array utilization in routine AN_SET.
AN_MFA	Next available column in filing array.
AN_MXZZ	Row number in filing array containing the column number of the successor entry. AN_MXZZ = AN_IMZZ + 1
AN_NOQ	Number of GASP files used.
AN_ISEED	Not used.
AN_MXX	Row number in filing array containing the column number of the predecessor entry. AN_MXX = AN_IMZZ + 2
AN_OUT	Integer variable indicating if an entry is to be stored (=0) or removed (=1) from the filing array.
AN_ATRIB(I)	Integer attribute transfer array used to enter contents of AN_ATRIB into a file entry or to extract the contents of a file entry into AN_ATRIB.
AN_INN(I)	Specifies the ranking rule for file I. A value of 1 means lovest value first, a value of 2 means highest value first.
AN_KRANK(I)	Specifies the attribute number of an entry in file I which determines ranking in the file.
AN_MFE(I)	This is a pointer to the column number of the first entry of file I.

AN_MLC(I)	This is a pointer to the column number of the next entry to be removed from file I.
AN_MLE(I)	This is a pointer to the column number of the last entry of file I.
AN_NQZZ(I)	This provides the number of entries currently in file I.
AN_TATRIB(I)	Integer array of same length as AN_ATRIB to be used as scratch storage. Typically used as buffer array in routines AN_FIND, AN_FIND2 and AN_FIND3 to hold a copy of the entry found.
AN_NSET(I,J)	Filing array to hold up to J entries of the GASP files. Rows 1 to I-2 are used for attribute storage, rows I-1 and I contain successor and predecessor pointers. While Figs. 4 and 7 show AN_NSET dimensioned with 500 columns, most models require only 100 to 200 columns.
AN_HROFST	Integer number specifying the hour at which the simulation is starting.
AN_MOFST	Integer number specifying the minute at which the simulation is starting. (i.e. at the start of simulation, AN_SIMTIM=0 corresponds to a model time of AN_HROFST hours and AN_MOFST minutes).
AN_TLAST	An integer variable containing the number of time units elapsed at the last event time since the start of simulation.
AN_TNEXT	An integer variable containing the event time of the next scheduled event in time units.
AN_TIMAX	An integer variable containing the maximum time in time units that will be set on the computer clock.
AN_SIMTIM	Integer variable to track the current simulation time in time units since the start of the simulation.
AN_SPEEDUP	An integer variable representing the model time speedup factor.
AN_TUPM	Time units per minute of model time. (i.e. AN_TUPM=10. means that 1 minute of model time corresponds to 10 time units as counted for example by AN_SIMTIM)

AN_EVENT	The number of the event to be executed next.
AN_ERR	Variable which is set to 0 if there is no command error.
AN_FLAG	General purpose variable, used for instance to determine type of initialization needed in the initialization module or to determine the type of interrupt received.
AN_MDLMOD	A variable indicating the model mode. AN_MDLMOD=1 specifies the synchronous mode while AN_MDLMOD=2 specifies the asynchronous mode.
AN_DATE(3)	Contains the date at the computer site at the start of a simulation run.
AN_TIME(2)	Contains the time at the computer site at the start of a simulation run.
AN_SWITCH(I)	A logical array element storing the status of ANEVENT switch I.
AN_DEVICE(I)	An integer array with elements equivalenced to PRINT1 - PRINT20. These array elements contain input or output device numbers for terminals, printers etc.
AN_INPBUF	An 80 character input buffer for reading operator commands.
AN_WORK(I)	A 10 element array mainly used for temporary storage of numbers in command line.
AN_SEED(I)	The current random number seed for random number stream I.
LINE_LEN	A variable which stores the length of the last line read.
AN_DIST_PAR(I,J)	A real array for storing distribution parameters.

8.0 CREATION OF THE ANEVENT LIBRARY

The ANEVENT library ANELIB contains all programs listed in Section 7.1 with the exception of the two main programs ANEVENT and ANEVENT1. This library must be recreated every time a change to the parameters or other dimension statements in file ANECOM.COB has been made. On the VAX this is accomplished by issuing the command:

```
@ANEBLD
```

This command will accomplish the following:

- Compile all subroutines listed in file SUBLST.COM to create object file SUBLST.OBI
- Assemble assembly program RN15.MAR to produce the object file RN15.OBI.
- Create a library ANELIB and insert all routines from files SUBLST.OBI and RN15.OBI.
- Delete all files with extension .OBI.

The contents of files ANEBLD.COM and SUBLST.COM are listed in Figures 5 and 6 respectively.

```
!       THIS PROCEDURE WILL CREATE THE LIBRARY ANELIB FROM
!       PROGRAMS LISTED IN FILE SUBLST.COM.
!       IF A DIFFERENT COMMON BLOCK SIZE IS REQUIRED,
!       FILE ANECOM.COB HAS TO BE CHANGED AND
!       THIS LIBRARY RECREATED.
!
$       FOR/OBJECT=SUBLST.OBI @SUBLST.COM
$       MAC/OBJECT=*.OBI RN15.MAR
$       LIB/CREATE ANELIB *.OBI
$       DELETE *.OBI.*
$       EXIT
```

Figure 5. Command procedure to create library ANELIB

```
ADD +-
CHANGE +-
CMDERR +-
CMNDRD +-
COMINT +-
COMTYP +-
CURPOS +-
DATAN +-
DEFILE +-
DIST +-
DSTROY +-
EVNTS +-
EVT001 +-
EVT002 +-
EVT003 +-
FILEM +-
FIND +-
FIND2 +-
FIND3 +-
HEADER +-
INFO +-
INITIA +-
INSCRN +-
INSERT +-
IRNG +-
LISTFL +-
MISCOM +-
MODRUN +-
QUIT +-
RAN1 +-
READLINE +-
RECALL +-
REMOVE +-
RMOVE +-
SAVEST +-
SET +-
SETIME +-
SIMDAY +-
SUBINT +-
SWITCH +-
TOSCRN +-
W2
```

Figure 6. SUBLST.COM list of subroutines in library ANELIB

9.0 DATA FOR COMMON BLOCK ANECOM

Initial values for some of the variables in the common block ANECOM are read in during initialization from a data file ANE.DAT. Figure 7 shows the contents of this data file for the common declarations shown in Figure 4. Note that the text portions represent comments which are not read by the program. Table 4 shows which routine and format is used for reading the various data items.

```
  11    3    4    6    3    6    5    5   98   99   77  8    9    3    2    6    6    6    5    6   AN_DEVICE
   T    F    T    F    F    T    F    F    F    F                AN_SWITCH
0,0,100000,0,1000,10.,9,0,    TLAST,TNEXT,TIMAX,SIMTIM,SPEEDUP,TUPM,HROFST,MOFST
1,    AN_MDLMOD
      5   1325       3       0       0       0       0       0       0       0   AN_SEED
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
      0       0       0       0       0       0       0       0       0       0
500,10,40,1179, AN_IDZZ,AN_IMZZ,AN_NOQ,AN_ISEED
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,  AN_KRANK
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,  AN_INN
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,
22,
1,5,208,1,8,0,3,3,1,2,0,
1,5,208,1,10,1,1,3,3,2,0,
1,10,208,1,10,0,1,2,2,1,0,
1,10,208,1,15,1,4,5,2,1,0,
1,40,208,1,40,0,4,1,3,8,0,
1,43,208,1,43,1,7,2,3,8,0,
1,50,298,50,1,0,0,0,0,0,0,
1,60,240,1,50,1,2,0,0,0,0,
1,77,240,3,100,1,1,0,0,0,0,
1,100,250,4,1,1,5,0,0,0,0,
1,150,208,1,160,0,10,1,1,5,0,
1,150,208,1,155,1,3,9,1,5,0,
1,155,250,3,2,0,5,0,0,0,0,
1,194,208,1,194,0,3,9,1,11,0,
1,194,208,1,194,1,5,1,1,11,0,
1,230,208,1,210,0,5,1,1,12,0,
1,230,208,1,210,1,6,8,1,12,0,
1,245,208,1,245,0,11,1,1,10,0,
1,245,208,1,245,1,4,6,1,10,0,
1,250,208,1,250,0,2,1,1,6,0,
1,251,208,1,251,1,4,1,1,6,0
1,255,208,1,255,1,2,1,1,17,0,
```

Figure 7. ANE.DAT data file for common block ANECOM

DATA ITEMS	ROUTINE WHICH READS THE DATA	FORMAT
AN_DEVICE	ANEVENT	20I4
AN_SWITCH	ANEVENT	20L4
AN_TLAST, AN_TNEXT, AN_TIMAX, AN_SIMTIM, AN_SPEEDUP, AN_TUPM, AN_HROFST, AN_MOFST	AN_COMINT	5I7, F12.6, 2I7
AN_MDLMOD	AN_COMINT	I7
AN_SEED	AN_COMINT	10I7
AN_IDZZ, AN_IMZZ, AN_NOQ, AN_ISEED	AN_DATAN	4I5
AN_KRANK	AN_DATAN	15I3
AN_INN	AN_DATAN	15I3
AN_DIST_PAR	AN_DATAN	5F10.2
JQ, AN_ATRIB[1]	AN_DATAN	11I7

[1] A JQ=-1 indicates the start of GASP file entry initialization, a JQ=0 terminates GASP file entry initialization. All intermediate entries specify the file number JQ and the attributes of the entries to be initialized.

Table 4. Common block initialization

10.0 TRANSFER OF ANEVENT PACKAGE TO ANOTHER COMPUTER

The ANEVENT package described here is implemented on a DEC/VAX digital computer and uses the VAX/VMS System Services routines and a machine-specific random number generator which may not be available on other computers. The following is a list of potential changes that may be required in order to implement the package on other computers.

10.1 Random Number Generation

Subroutine AN_RAN1 calls a uniform random number generating subroutine AN_RN15 which is coded in assembly language. AN_RN15 produces uniformly distributed numbers between 1 and 32767 using an integer seed as argument.

10.2 Operator Interrupt and Time Keeping

VAX/VMS System Services routines have been used to handle operator interrupts and time keeping. These routines are used in the program as follows

ANEVENT	lines	60 to 63	Variable type and data declarations
		67 to 69	Declare an event flag cluster starting with flag 64.
		71 to 73	Determine the name of the terminal used by the operator.
		89 to 96	Flag 66 is used to detect a model operator interrupt, flag 65 is used to indicate a timer interrupt. Both flags are cleared.
		98 to 100	The operator terminal is assigned to the QIO channel AN_CHAN.
		170 to 173	The model operator flag is cleared.

AN_MODRUN	lines	72 to 73	Variable type and data declarations. MASK is set to 6 to allow a scan of event flags 65 to 66 in the cluster starting with flag 64.
		85 to 105	Wait for either flag 65 or 66 to be set, then read the flag status FLAGSET, cancel the timer and QIO and screen the flag status using the mask MASK.
		119 to 120	Clear the timer flag
		126 to 127	Clear the model operator flag
AN_QUIT	lines	59 to 60	Release the event flag cluster
AN_SETIME	lines	629 to 632	Variable type and data declarations
		646 to 650	Set the QIO for the operator terminal, channel AN_CHAN, flag AN_OPFLAG.
		652 to 654	Set the timer to a value TIMER and assign flag AN_TIMFLG to the timer.

10.3 Format Descriptor $

The format descriptor $ is used to suppress carriage return when prompting user for a response. It can be found in

AN_CHANGE	line	100	to prompt for attribute number and value
AN_CMNDRD	line	72	to prompt for command type
AN_CURPOS	line	12	to store or restore cursor position
AN_INFO	line	77	to prompt for GASP file
	line	106	to prompt for Histogram

AN_INITIA	line	70	to prompt for a cold start
	line	85	to prompt for input journal file
	line	90	to prompt for recall time
	line	99	to prompt for output journal file
	line	103	to prompt for nev journal file
	line	108	to prompt for time interval for saves
AN_INSCRN	line	10	to erase screen
	line	13	to set scrolling region
AN_MISCOM	line	169	to prompt for switch group
	line	191	to prompt for interrupt schedule
AN_READLINE	line	109	to vrite input journal file line
AN_SWITCH	line	68	to prompt for switch setting
AN_SUBINT	line	59	to prompt for alphanumeric display
	line	65	to prompt for display update interval
AN_TOSCRN	line	14	to select normal video
	line	17	to position cursor

10.4 Reading Time of Day and Date

In ANEVENT lines 85 and 86, calls to subroutines DATE and TIME are made to obtain the date and time which are stored in AN_DATE and AN_TIME respectively.

11.0 BIBLIOGRAPHY

PRITSKER, A.A.B. and KIVIAT, P.J.
1969. Simulation with GASP II, a Fortran Based Simulation Language. Prentice-Hall Inc., Englewood Cliffs, N.J.

PHAN, C.S.
1979. Generation of Random Samples Using Non-Parametric Methods. Division of Mechanical Engineering, National Research Council, Ottawa, Ontario Report No. LTR-CS-199.

CRATE, G.F. Ltd.
1982. ANEVENT: An Interactive Computer Modeling Package Based on Discrete Event Simulation - A User's Guide. Division of Mechanical Engineering, National Research Council, Ottawa, Ontario Report No. LTR-AN-53.

PRITSKER, A.A.B.
1974. The GASP IV Simulation Language.
John Wiley and Sons Inc. New York, N.Y.

REPORT DOCUMENTATION PAGE/PAGE DE DOCUMENTATION DE RAPPORT

REPORT/RAPPORT	REPORT/RAPPORT
1a TR-SYS-020	1b NRC 29530
SECURITY CLASSIFICATION/ CLASSIFICATION DE SÉCURITÉ Unclassified 2	DISTRIBUTION/DIFFUSION ☐ Controlled/Contrôlé ☒ Unlimited/Illimité 3

TITLE/SUBTITLE/TITRE/SOUS-TITRE

4 ANEVENT An Interactive Computer Modeling Pacakge Based on Discrete Event Simulation (Update of LTR-AN-48)

AUTHOR(S)/AUTEUR(S)

5 G.H. Beland, A.W. Chan, P.W.U. Graefe, L.K. Nenonen

SERIES/SÉRIE

6

CORPORATE AUTHOR/PERFORMING AGENCY/AUTEUR D'ENTREPRISE/AGENCE D'EXÉCUTION

7

SPONSORING AGENCY/AGENCE DE SUBVENTION

8

DATE	FILE/DOSSIER	LAB. ORDER COMMANDE DU LAB.	PAGES	DIAGS	REFS
9 1988/08	10	11	12a 38	12b 7	12c 4

NOTES

13

DESCRIPTORS(KEY WORDS)/MOTS-CLÉS

14

SUMMARY/SOMMAIRE 15 ANEVENT is a collection of FORTRAN based subroutines for building interactive discrete event simulation models. The overall structure of such models, both interactive and non-interactive, is described. The purposes and methods of use of the routines in the package are also given.

ADDRESS/ADRESSE 16 Dr. Vince Thomson Tel: (613) 993-9461
Head
Systems Laboratory
Division of Mechanical Engineering
National Research Council Canada
Montreal Road, Ottawa, Ontario K1A 0R6